CORRECTED VERSION*

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/30, 9/46

A1

(11) International Publication Number: WO 00/02148

(43) International Publication Date: 13 January 2000 (13.01.00)

(21) International Application Number: PCT/US99/15091

(22) International Filing Date: 2 July 1999 (02.07.99)

(30) Priority Data:
09/109,426 2 July 1998 (02.07.98) US

(71) Applicant: INTERLEAF, INC. [US/US]; 62 Fourth Avenue, Waltham, MA 02451 (US).

(72) Inventor: REISTROFFER, Kirk, L.; Apartment #30, 919 Mowry Avenue, Fremont, CA 94536 (US).

(74) Agents: LAURENTANO, Anthony, L. et al.; Lahive & Cockfield, LLP, 28 State Street, Boston, MA 02109 (US).

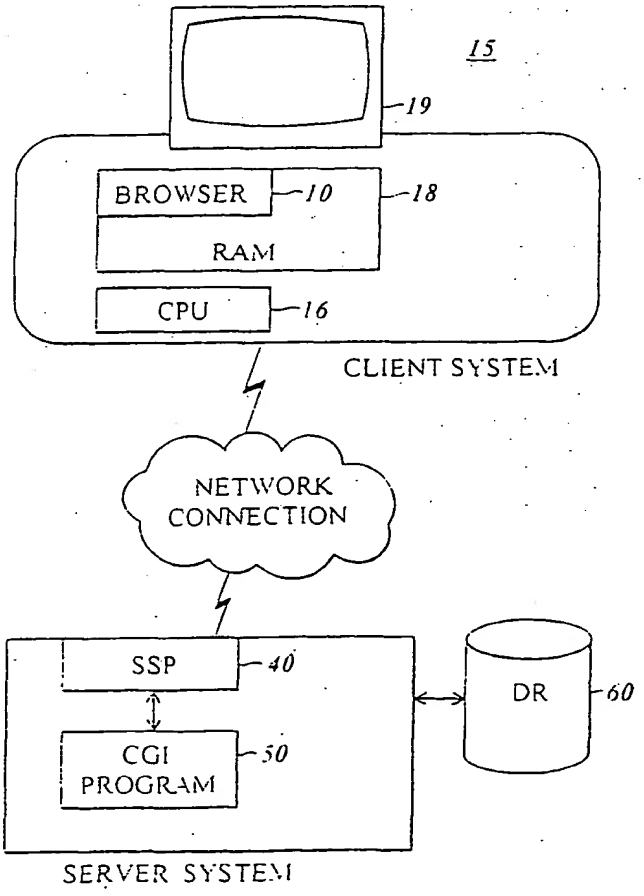
(81) Designated States: CA, JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SYSTEM AND METHOD FOR RENDERING AND DISPLAYING A COMPOUND DOCUMENT

(57) Abstract

The present invention provides for the rendering and displaying of a compound document at a client system without requiring that any application specific-software be installed on the client system. The application is said to have a "zero footprint" since the software that renders and displays the compound document resides only temporarily on the client system, and is not persistently or permantently stored when the rendering and displaying application is not active. The compound document is also not persistently or permanently stored upon termination of the application.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

SYSTEM AND METHOD FOR RENDERING AND DISPLAYING A COMPOUND DOCUMENT

Field of the Invention

The current invention relates generally to a system and method for rendering and displaying a compound document, and more particularly to a system and method for rendering and displaying a compound document using an applet within a client/server architecture.

Background of the Invention

Access to a large volume of documentation is a critical requirement in many industries. For example, in the aircraft industry the development of a new aircraft requires that the engineering specifications for perhaps over one million parts must be made readily available to those who need the documentation. In another example, the documentation for a company's computer systems and software must be accessible by the software engineers who program and operate the company's computer systems. In both these examples the documentation that must be disseminated includes text, graphics, and possibly images, rather than just text.

Because of the large number of people that must have access to this documentation in these applications, it is impractical to provide hard copy versions of the documentation to everyone or to a substantial subset of the people. Additionally, any modification to the documents requires regeneration of the hard copies of the documents. As a result, this method of distributing hard copies of the documents is both time consuming and costly. Moreover, there is a risk that the documents that have been distributed are out of date due to the inherent lag time in making modifications to the documentation.

In recent years with the prevalence of compact disk-read only memory (CD-ROM) drives in personal computer systems, CD-ROMs have been used as a means for disseminating large volumes of documentation. In particular, the documents are stored on CD-ROM, and the CD-ROMs are distributed to the interested parties. When the documentation is updated, a new CD is generated and disseminated with the update.

The use of CD-ROM thus has eliminated the manual process of updating a hard copy of the document at the cost of the increased expense of generating a new CD-ROM for each update or batch of updates to the documents. The use of CD-ROM does not address, however, the risk of the data being out of date since there is still a lag time in distributing the CD-ROM.

Some attempts to find improved methods for the distribution of large volumes of information have focused on the use of computer systems employing a client/server architecture. In this architecture a client computer system, typically a personal computer or workstation, is remotely connected to a server computer system that has access to a central document database. In one approach, the client system includes software ("client-side software") that is permanently installed on the system for accessing documents from the document database. The client-side software acts in concert with cooperating software at the server ("server-side software") to permit distribution of documents from the central document database. Employing a central database in this architecture addresses issues regarding the timeliness of the data and efficient updating of the data, since the documentation is updated only once and then is available to all the client systems. Nevertheless, there are still certain disadvantages in this approach. The client-side software is permanently installed on the client's system. Thus, any changes or modifications to the client-side software require a new installation of the software at the client system in order to implement the changes. In a large network this may necessitate updating the client-side software at thousands of personal computers which is time consuming. This process is potentially complicated because the client-side software and server-side software are cooperating applications, and synchronization must be maintained. Updates to the server-side software cannot be applied until all the client systems are updated to reflect the new changes. In a worst case scenario, both the client-side software and the server-side software must be updated as one event to insure operability. This can impact the availability of the central document database. Furthermore, system security may be reduced since the application is permanently installed on the client system where the application may be altered to provide unauthorized capabilities. A modified version of the application may be substituted for the original version, without the server-side becoming aware of it

More recently, with the explosion in the use of the Internet and intranets, a new architecture has found favor in connecting client systems with servers. In this new architecture, a web browser is run on the client system, providing the user with access to a server over the Internet or intranet. The browser is not an autonomous client application but rather operates in cooperation with instructions from the remote server software. To be more specific, once a specific server has been invoked by the browser, the content that the browser displays on a computer device is at the direction of the server-side software. The server communicates with the browser in a language termed HyperText Markup Language, or HTML. This language facilitates the specification of HyperText documents. In a typical communication between the browser and the server, HTML files are sent from the server to the browser to direct the operation of the browser. The browser receives the HTML language, interprets it, and performs the operations called for by the HTML. Communication back to the server from the browser is in the form of the HTTP protocol. In networks where HTML files are passed between client systems and server systems, the client/server architecture is maintained, but the client-side software is a generic browser rather than an autonomous or permanently installed software for interpreting and displaying a document from the server. These generic browsers can interpret and display the HTML markup language. Unfortunately, HTML is a text only language and does not support compound documents combining text, graphics and images that are self contained within the document without the need to reference other files or documents. Additionally, an HTML document is transferred only as a single file, and there is no capability to view only specific pages of the document or certain sections of the document independent of the entire document. For very large documents this increases the transmission time and requires the document to be navigated from the beginning of the document which is of course awkward and frustrating when only a small portion of a very large document needs to be reviewed.

It is thus an object of the invention to provide for a document viewer at a client system that handles text, graphics, and images, but is not persistently or permanently installed on the client.

It is a further object of the invention to provide for a document viewer at a client system that handles compound documents where the documents are not persistently available on the client system.

Summary of the Invention

The present invention provides for the rendering and displaying of a compound document at a client system without requiring that any application specific-software be installed on the client system. The application is said to have a "zero footprint" since the software that renders and displays the compound document resides only temporarily on the client system, and is not persistently or permanently stored when the rendering and displaying application is not active. The compound document is also not persistently or permanently stored upon termination of the application. Consequently, document and application security is enhanced through the practice of the invention. A compound document is a self contained document structure which has the capacity to include, in combination, text, graphics, and images. By self contained, it is intended that the text, graphics, and images of the document are fully described within the document without reference to external documents.

In one aspect of the invention, the rendering and display of a compound document is initiated at a client system through a Java-capable web browser. The web browser is capable of communicating to a server system through a network connection and operates on HTML formatted data. By "Java-capable" is meant the ability of the web browser to recognize an APPLET or OBJECT tag and facilitate the loading and running of the applet at the client system.

In another aspect of the invention, the compound documents that are to be displayed at the client system may reside in one or more document repositories that are remote to the client system, but accessible from a server system that is in communication with the client system. The server system includes software for accessing and manipulating a compound document, and is capable of compressing the compound document. Furthermore, the software is capable of scaling images within the compound document to accommodate the resolution requirements of the client system. For example, high resolution documents may be scaled down to lower resolutions to

accommodate lower resolution display devices at client systems which also advantageously reduces the data transmission load the memory requirements of the client system.

In another aspect of the invention, a Java applet is downloaded upon demand to the client system to render and display the requested compound document. The Java applet only resides on the client system for the time it is active and is not persistently or permanently stored on the system.

In yet another aspect of the invention, the Java applet includes a graphical user interface (GUI) for controlling the presentation of the compound document at the client system. The GUI advantageously permits navigation of the document by page numbers and hypertext link traversal.

According to another aspect, the invention employs a method for displaying a compound document residing on a server system at a client system. The client system includes a processor for executing computer instructions and which is capable of communicating with the server system. The method comprises the steps of requesting the compound document from the server system by the client system, receiving the compound document and first computer instructions corresponding to commands for displaying the compound document, generating in response to the first computer instructions at the client system second computer instructions executable by the processor of the client system for displaying the compound document, executing the second computer instructions at the client system, and displaying the compound document.

According to another aspect, the method further comprises the steps of providing browser software instructions at the client system, and executing the browser software on the client system for specifying the requested compound document.

According to another aspect, the first and second computer instructions are not persistently stored on the client system. According to still another aspect, the method also provides for a compound document having text, image, or graphic data, and provides for scaling the compound document at the server system to adjust the display of the compound document at the client system.

The invention also provides for a computer readable media having computer instructions thereon corresponding to commands for displaying a compound document received at a client system, the client system having a computer processor and is capable of executing browser software thereon. The computer instructions including first computer instructions generated by compiling source statement instructions, thereby causing the browser software, coupled to a JAVA Virtual Machine, to generate second computer instructions executable by the processor of the client system for displaying the compound document.

According to one aspect, the first computer instructions are JAVA bytecode, and the source statement instructions are instructions from the JAVA programming language.

According to another aspect, the compound documents consist of information from the group of text, image, and graphic data, and according to one practice includes text and image data.

The present invention also provides for a document distribution or display system residing on a server system for transferring a compound document to a client system for display thereon. The document distribution system includes means for receiving a request for transferring the compound document to said client system; first transfer means for transferring pseudo computer instructions from said server system to said client system corresponding to computer instructions that are executable at said client system to display said compound document; and second transfer means for transferring to said client system said compound document in response to said executable computer instructions at said client system.

According to one aspect, the document distribution system also includes means for compressing the compound document prior to transmitting the compound document, or means for scaling the compound document for display at the client system.

According to another aspect, a document display system, executable on a client system, for displaying at the client system a compound document residing on a server system, includes means for requesting the compound document from the server system, means for receiving from the server system the compound document and first computer instructions indicative of the second computer instructions for displaying the compound

document at the client system, and means for generating in response to the first computer instructions the second computer instructions. The second computer instructions are executable by a processor of the client system for displaying the compound document thereat. The system also includes structure for executing the second computer instructions at the client system, and displaying the compound document.

Brief Description of the Drawings

An illustrative embodiment of the present invention will be described below relative to the following drawings.

FIG. 1 is a depiction of a computer architecture, including the major software components suitable for practicing an illustrative embodiment of the current invention.

FIG.2A depict a sample screen display for selecting a compound document at the client system.

FIG 2B depicts an alternative screen display to FIG. 2A for selecting a compound document at the client system.

FIG. 3 illustrates components of a Web browser, the Java applet, and JVM residing in RAM of the client system.

FIG 4A illustrates the steps that are taken to initiate a document rendering and display application according to the illustrative embodiment of the present instant invention.

FIG 4B illustrates the steps that the server-side software initially takes to respond to the request for a compound document at the client system.

FIG 5A illustrates the steps that the Web browser takes after receiving the HTML data file describing the selection page.

FIG. 5B illustrates the steps that the server-side software takes in responding to the compound document request by the client system.

FIG 6A illustrates the steps that the server-side takes after receiving the HTML data file identifying applet and the compound document to be displayed.

FIG 6B shows the steps that applet 70 takes after receiving control.

FIG. 7 shows an example of a graphical user interface (GUI) suitable for practicing the illustrative embodiment of the invention.

Detailed Description of an Illustrative Embodiment

The present invention provides for an innovative system and method for rendering and displaying a compound document at a client system without requiring that any application specific software be installed on the client system. The application is said to have a "zero footprint" since the software that renders and displays the compound document resides only temporarily on the client system, and is not persistently or permanently stored when the rendering and displaying application is inactive. An applet is advantageously utilized to render and display the compound document without requiring that the document be sent in the HTML format to the browser.

FIG. 1 illustrates a compound document delivery system with a combination of resources including client and server systems which may be personal computers, diskless network computers, palmtops, or workstations as clients, and workstations, minicomputers, or mainframes as servers. The various elements may be coupled to one another by a LAN (Local Area Networks) or a WAN (Wide Area Network), and also provide access to the Internet in what is typically termed the World Wide Web.

It will be seen from FIG. 1 and FIG. 3 that this illustrative embodiment of the invention includes a Web browser 10 on client system 15 which is coupled to a Web server system 20 via a network connection. Client system 15 includes a central processing unit (CPU) 16 for executing computer instructions, random access memory (RAM) 18 for temporarily storing data, a display device 19 or monitor for displaying a

compound document, and a point-and click device 17 such as a mouse for making selections. A means for permanently storing data such as a hard disk may optionally be included. The network connection typically employs a TCP/IP communication protocol. Web browser 10 is a software program that executes in the random access memory 18 of client system 15. There are at present a number of commercially available Web browsers that are suitable to practice the invention. Examples of such Web browser software programs are Navigator and Communicator from Netscape, Inc. of Mountain View, California and Internet Explorer (IE) from Microsoft Corporation of Redmond, Washington. Web browser 10 executes in concert with an operating system on a client system 15 which may be a personal computer, a UNIX based computer system, or a diskless network computer.

Web browser 10 is computer software for receiving, interpreting and displaying information coded in HTML. HTML (HyperText Markup Language) is the language used by Web servers 20 for encoding documents that are viewed by a Web browser 10 at client system 15. The specification for the HTML language may be obtained on the Internet at the Sandia Laboratories Web site. The URL address is www.sandia.gov.

Web server 20 is coupled to client system 15 via a network connection, and may also be coupled to other server systems and computers 30 by an Internet connection or a wired telecommunications line or a wireless connection. Web server 20 includes a server software program (SSP) 40 for sending and receiving data to and from Web browser 10 on client system 15. SSP 40 is capable of communication with a CGI PROGRAM 50. In the preferred embodiment of the invention, CGI PROGRAM 50 physically resides on Web server 20 or is distributed on a number of servers with at least one identified component on web server 20. In one practice of the invention, SSP 40 communicates with CGI PROGRAM 50 through CGI (Common Gateway Interface) scripting, a Netscape Application Programming Interface (NSAPI) or Microsoft's Internet Server Application Programming Interface (ISAPI). To those of ordinary skill in the art, CGI scripting, NSAPI, and ISAPI are well known industry standards for interfacing a server program 40 and an application specific computer program such as CGI PROGRAM 50.

Web browser 10 is capable of receiving and interpreting a Java applet. Java is an object oriented programming language developed by Sun Microsystems of Mountain View, California. Modeled somewhat after C++, the Java language is designed to be portable across client systems and operating systems. To be Java-capable means that the browser can download and run Java programs called applets on the client system. Applets are programs that are downloaded over the World Wide Web by a web browser and run inside an HTML web page. An applet is created using the Java language and compiled using a Java compiler. The HTML text markup language definition includes an APPLET tag or OBJECT tag which contains a reference to a Java applet on the server. In the following discussion, this illustrative embodiment will be described in terms of the APPLET tag, but one of ordinary skill in the art will recognize that the discussion is also relevant and applicable to the use of the OBJECT tag to identify an applet. An applet is referred to in an HTML page by reference to the APPLET or OBJECT tag, and the HTML and Java files are stored at a web site in the same manner as ordinary HTML files. When a Java enabled browser encounters the embedded APPLET tag, the browser downloads the applet to the client system. Java enabled Web browsers pass the APPLET tag to their built-in Java Virtual Machine (JVM) for handling. A Java Virtual Machine is software that controls the execution of the applet. The JVM makes requests back to the server to fetch the Java bytecode for the applet, loads the bytecode into the client's random access memory and optionally compiles it into client native machine code. Bytecode is a pseudo instruction set that corresponds to source Java statements. The bytecode is machine independent, and is optionally translated into the client system's native machine code before execution. The use of a bytecode insures platform independence. The JVM causes the applet code to execute.

In a preferred practice of the invention, CGI PROGRAM 50 is embodied as Xtravert, a software product developed by Interleaf, Inc. of Waltham, Massachusetts. CGI PROGRAM 50 provides auxiliary services for the processing of compound documents from a document repository (DR) 60 or collection of linked files. A compound document is a document that may contain a combination of text, graphics and images within the document. The preferred structure for a compound document is described in U.S. Patent 5,579,519 entitled EXTENSIBLE ELECTRONIC

DOCUMENT PROCESSING SYSTEM FOR CREATING NEW CLASSES OF ACTIVE DOCUMENTS, assigned to Interleaf, Inc. and incorporated by reference herein.

CGI PROGRAM 50 is capable of accessing compound documents or portions of compound documents. SSP 40 passes the compound document through CGI PROGRAM 50 in the process of sending the document to client system 15. CGI PROGRAM 50 may manipulate or modify the document in order to accommodate any specific limitations of the client system or to provide enhanced functionality. As an example, images imbedded within the document may be scaled from high resolution to low resolution to adjust for specific rendering limitations at the client system. One of ordinary skill in the art will recognize that GCI PROGRAM 50 is not required to practice the invention, but is employed to provide additional functionality such as compression and scaling at the server.

The method for requesting and rendering a compound document from a client system in accordance with the illustrative embodiment is now described. Web browser 10 is launched at client system 15 as was described earlier. Web browser 10 is any web browser capable of running a Java applet. Web browser 10 allows communication to and interaction with other server-side applications through the HTTP protocol using an HTML formatted file to encapsulate the data content. A user at client system 15 who desires to connect to the display application enters a HTTP command into web browser 10, thus directing the browser to make contact with the server-side software over a network connection. An example of a suitable H TTP command for this purpose and is "http://www.serverapp.com/foo.html", where "serverapp" is the name of the application on the server. SSP 40 receives the HTTP command, and sends back to the browser via SSP 40 an HTML document stream that is rendered on client system 15 by web browser 10 to provide the ability to choose a document to display.

As shown in FIG. 2A, and for illustrative purposes only, the HTML stream is rendered at web browser 10 to display a screen allowing the user to select a document to be displayed at client system 15. Alternatively, as shown in FIG. 2B, the user may be presented with a list of documents and asked to choose one for display by using point and click device 17. Having chosen a document for display, SSP 40 is notified of that

choice by client system 15 through another HTTP request. Accordingly, SSP 40 creates a HTML file and sends the file to web browser 10. The HTML file notifies Web browser 10 that an applet must be loaded into the RAM 18 of the client system 15 by including an APPLET tag in the data stream which identifies the Java applet that is required for rendering and displaying the compound document at client system 15.

An example of an HTML page containing an APPLET tag follows:

```
<title>Interleaf Xtravert Example</title>
<hr>
<APPLET CODE="COM/interleaf/iv/Xtravert.class" WIDTH=660
HEIGHT=450>
<PARAM NAME="plURL" VALUE="/foo.pl">
<PARAM NAME="imgCGI" VALUE="/cgi-bin/wvj_plimgopt">
Sorry, your Web Browser isn't Java enabled.
</APPLET>
<hr>
```

FIG. 3 illustrates Web browser 10, applet 70, and JVM 80 residing in RAM of client system 15. When the applet 70 is transferred to web browser 10, web browser 10 recognizes the "<APPLET " tag and the "WIDTH=660 HEIGHT=450>" command. Web browser 10 reserves a 660 x 450 pixel area on the HTML page's display area for the applet 70 and passes everything from the beginning "<APPLET " tag through the ending </APPLET>" tag to its embedded JVM 80. The display area for the applet 70 is variable and can be set to any appropriate value. JVM 80 uses the "CODE=" field of the APPLET tag to get the server location of the applet 70 and it requests the top level COM/interleaf/iv/Xtravert.class" from the server. In this example the applet 70 is the Xtravert applet 70, which has been previously described as the preferred embodiment of the invention. In loading that class into client memory, the JVM 80 may determine other classes that are needed and request them too. The JVM 80 then starts the applet 70 and passes the applet 70 the parameter "/foo.pl" as the location on the server of the compound document that applet 70 is to display in the 660 x 450 pixel space. The JVM

80 also passes the applet 70 the parameter "/cgi-bin/wvj_plimgopt" indicating the location on the server of CGI PROGRAM 50 to scale images in the foo.pl document to the applet 70's current display resolution, as the document is being streamed from server to client.

When the applet 70 initiates (i.e. begins execution), the applet 70 requests from the server the compound document's data stream by issuing an HTTP request to the server identifying the document to be transferred, the scaling parameter as discussed further below, and the CGI interface for scaling and compressing the compound document prior to sending the document to the applet. Applet 70 optionally may request only a subset of the document rather than the complete document. To request a subset of the compound document applet 70 provides a parameter in the HTTP request to notify the CGI interface CGI PROGRAM 50 of the specific pages or page that is requested. As was previously discussed, CGI PROGRAM 50 on server 20 appropriately scales the images according to the characteristics of client system 15, and the performance considerations. Applet 70 instructs CGI PROGRAM 60 to scale images according to one of the following methods:

1. Decimation - elimination of pixels from the image according to the scaling factor that is desired. For example, to scale an image down by a factor of two, every other pixel must be eliminated.

2. Replication - Replication of pixels from the image according to the scaling factor that is desired. For example, to scale an image up by a factor of two, every pixel is repeated.

3. Image Convolution - Area averaging of pixels to generate a representative pixel. For example, two pixels are averaged to generate a representative pixel.

4. Error Distribution - Area averaging with the error difference distributed to adjacent pixels.

SSP 40 retrieves the compound document and optionally passes the document to CGI PROGRAM 50 which compresses the compound document before passing it back to SSP 40 for distribution to the web browser 10. Compression of the compound document by CGI PROGRAM 50 reduces the transmission time of the document, and thus provides for a faster response. The applet 70 upon receipt of the document decompresses the compound document prior to rendering and displaying it

The compound document is sent by SSP 40 to applet 70 in the compound document's native file structure, including the compression and scaling performed by CGI PROGRAM 50. In other words, the present invention advantageously eliminates the need to wrap the compound document in an HTML format in order to send it to the client system 15. Additionally, there is no need to define special HTML tags to transfer the compound document. The applet 70 reads the document's first page from the received transmission stream and renders it for display on the client's monitor 19, either in place on the browser's original HTML page display or in a separate window.

In one advantageous aspect of the invention, the user's intellectual property (i.e. the user documents) are protected from modification, while providing access to the documents. The compound document at DR 60 is never modified, since only a copy of the document is delivered to client system 15 and the original is not altered. Furthermore, since the compound document is not persistently stored on the client system 15, there is a reduced possibility of unauthorized access to the compound document. It is clearly seen that the present invention has improved document security over conventional systems that permanently store the document on client system 15.

A significant feature of the present invention is that the applet 70 may render and display other compound documents requested from DR 60 while the applet 70 is active and is currently executing in the memory of client system 15 under the auspices of web browser 10. These additional requests are made in the same manner as the first by executing an http request to the server system 20. When applet 70 is terminated, the applet 70 and compound document are not persistently stored on the system. It is said that the applet 70 has a "zero footprint" in that the applet 70 does not permanently reside on the client system 15, but rather is loaded from server 20 when required and is only

resident on client system 15 during execution of the applet 70. At no time is the applet 70 installed on client system 15.

Since the applet 70 is always loaded from server 20 upon request and there is no installed portion of the document rendering and display application permanently resident on client system 15, the need to install application specific software on client system 15 is eliminated. The applet 70 can be modified at the server 20, compiled at the server 20, and is immediately available for download to the client systems 15 without modification to those systems.

FIG 4A illustrates the steps that are taken to initiate a document rendering and display application according to the instant invention. Initially, a Java capable web browser is launched at client system 15, step 410. A user invokes the document rendering and display application on server 20 through use of the HTTP protocol by entering "http://www.serverapp.com/foo.html" at web browser 20, step 420. Web browser 10 sends the HTTP request to server 20, step 430.

FIG 4B illustrates the steps that the server-side software initially takes to respond to the request for a compound document at client system 15. SSP 40 receives the HTTP protocol request, step 440. SSP 40 accesses the HTML data file foo.html that provides the user the ability to specify a compound document for display at client system 15, step 450 and sends it to Web browser 10, step 460

FIG 5A illustrates the steps that the Web browser 10 takes after receiving the HTML data file describing the selection page, foo.html. Web browser 10 initially interprets the HTML data file to create a display screen at client system 15, step 510 , and displays it on the monitor 19 of the system, step 520 . After the user selects a compound document to be displayed or a portion of a compound document such as a page, step 530. Web browser 10 sends the response to SSP 40 by the HTTP protocol, step 540 .

FIG. 5B illustrates the steps that the server-side software takes in responding to the compound document request by client system 15. SSP 40 receives the HTTP request from client system 15, step 550 An HTML file that includes an APPLET or OBJECT tag identifying the applet 70 to be loaded and also the location of the compound

document that is to be displayed is generated and shown as step 560 . SSP 40 delivers the HTML file, including the imbedded APPLET tag, to Web browser 10, step 570 .

FIG 6A illustrates the steps that the client-side takes after receiving the HTML data file identifying applet 70 and the compound document to be displayed. Web browser 10 renders the HTML data file, step 600 and identifies the APPLET or OBJECT tag, step 610 . Web browser 10 requests that applet be loaded from server 20, step 620 and sizes the presentation window for display of the compound document, step 630 . Upon receiving in RAM 18 the bytecode of applet 70, Web browser 10 passes the bytecode to JVM 80 further processing, step 640 . JVM 80 may either interpret the bytecode or generates native machine code for execution of applet 70, step 650 . Control is passed to applet 70, step 660 .

FIG 6B shows the steps that applet 70 takes after receiving control. Applet 70 requests the compound document from server 20 by issuing an HTTP request to the server identifying the compound document as a /f00.pl parameter and optionally a pointer to a CGI PROGRAM for providing any additional document support, step 670. Upon receipt of the document, applet 70 renders and displays the document on client system 15, step 680. Applet 70 may optionally request additional compound documents for further display, step 690 .

As has been previously discussed, applet 70 is never installed on client system 15, but rather is temporarily stored in the RAM of client system 15 only during the time it is active. Upon termination of applet 70, both applet 70 and the compound document are not persistently or permanently stored on client system 15. The process advantageously has a "zero footprint".

In one embodiment of the invention, the compound documents that are rendered and displayed by applet 70 contain graphics described by vector illustrations. Vector illustration is a graphic technique for representing an object as the combination of filled-in polygons. The polygons, which are not restricted by size or the number of sides, in combination define the shape of the object. Each polygon may be filled-in or colored independently of the other polygons. Because of these features vector illustration provides a powerful technique for representing a graphical object with a high degree of granularity. The compound documents as described in U.S. Patent 5,579,519

previously incorporated by reference and contemplated herein may include vector illustrations to represent an object. Unfortunately native Java does not provide the ability to interpret complex vector illustrations. Applet 70 includes support for interpreting and rendering a graphical object represented by complex vector illustrations. Complex vector illustration is advantageously used to represent Japanese characters, and can be further used to support character sets not supported by native Java.

In another embodiment according to the instant invention, the cooperative interaction between the client-side software and the server-side software allow for the manipulation of images in order to accommodate the client system 15, and support additional functionality that is lacking in native Java. CGI PROGRAM 50 on server 20 can alter the presentation of an image within a compound document for display at client system 15 by changing the compound document to reflect the changes. As has been previously described, CGI PROGRAM 50 can scale the resolution of images within a compound document to accommodate the capabilities of client system 15. Further, CGI PROGRAM 50 can manipulate an image within a compound document to cause rotation of the image during the presentation of the image. This feature is particularly useful in support of a landscape type display of the compound document. Scaling and rotation of images is necessarily performed by the server-side software, since these features are not supported by the functionality of JVM 80. Compression techniques are employed on the server-side to reduce network transmission time and client memory costs in sending high resolution image data to a low resolution client

In another practice of the invention, applet 70 includes a graphical user interface (GUI) as shown in FIG. 7 for controlling the presentation of the compound document at client system 15. Referring to FIG. 7, the GUI includes a presentation window 705 and a toolbar 710 consisting of one or two rows of "controls" (i.e. menus or buttons) manipulating the contents of presentation window 705. The first row of "controls" is the history bar 715 which allows the traversing of previously displayed documents by following HyperText links. History bar 715 includes a scrollable history 720 of the links and buttons for moving forward 725 and backward 730 in the link history. The history bar 715 is initially invisible, and is made visible the first time that a user follows a link and remains visible while applet 70 is active. The second row 735 of the toolbar is

always visible, and controls the presentation of the current compound document in the presentation window 705. Row 735 includes a FIRST PAGE 740 button for presenting the first page of a compound document, a NEXT PAGE 745 button, a PREVIOUS PAGE 750 button, and a LAST PAGE 755 button. These buttons operate in a conventional manner. A new window can be sized within the original presentation window 705 by selecting the NEW WINDOW 760 button to permit the display of an additional compound document by applet 70. This additional window permits the simultaneous display of two or more compound documents. Row 735 also includes a REFRESH 765 button for refreshing presentation window 705, a PRINT 770 button for printing the compound document, and a HELP 775 button. Row 735 also includes a selection box 780 for "zooming" in or out of a compound document. Presentation window 705 can be scrolled vertically through a VERTICAL SCROLL BAR 785 and horizontally through a HORIZONTAL SCROLL BAR 790.

Having described the invention, what is claimed as new and desired to be secured by Letters Patent is:

What is claimed:

1. A method for displaying a compound document residing on a server system at a client system, said client system having a processor for executing computer instructions and capable of communication with said server system, the method comprising the steps of:

requesting said compound document from said server system by said client system;

receiving from said server system said compound document and first computer instructions corresponding to commands for displaying said compound document;

generating in response to said first computer instructions at said client system second computer instructions executable by said processor of said client system for displaying said compound document;

executing at said client system said second computer instructions, and

displaying said compound document at said client system.

2. The method of claim 1, further comprising the step of providing browser software at said client system; and wherein said requesting step further includes the step of executing said browser software on said client system for specifying the requested compound document.

3. The method of claim 1, further comprising the step of providing a compound document having information selected from the group of text, image, and graphic data.

4. The method of claim 1, wherein said first and said second computer instructions are not persistently stored on said client system.

5. The method of claim 1, further comprising the step of scaling said compound document at said server system to adjust the display of said compound document at said client system.

6. The method of claim 2, wherein said receiving step further includes receiving JAVA bytecode by said browser at said client system.

7. The method of claim 2, wherein said requesting step further includes initiating by said browser software a request for the transmission of said compound document using the HTTP protocol.

8. The method of claim 3, wherein said displaying step further includes the step of rotating said graphic data by said browser software before displaying said graphic data on said client system

9. A computer readable media having computer instructions thereon corresponding to commands for displaying a compound document received at a client system, said client system having a computer processor and capable of executing browser software thereon, said computer instructions including first computer instructions, generated by compiling source statement instructions, causing said browser software, coupled to a JAVA Virtual Machine, to generate second computer instructions executable by said processor of said client system for displaying said compound document.

10. The computer readable media of claim 9, wherein said first computer instructions are JAVA bytecode.

11. The computer readable media of claim 9, wherein said source statement instructions are instructions from the JAVA programming language.

12. The computer readable media of claim 9, wherein said compound documents consist of information from the group of text, image, and graphic data..

13. A document distribution system residing on a server system for transferring a compound document to a client system for display thereon, said document distribution system comprising;

means for receiving a request for transferring said compound document to said client system;

first transfer means for transferring pseudo computer instructions from said server system to said client system corresponding to computer instructions that are executable at said client system to display said compound document; and

second transfer means for transferring to said client system said compound document in response to said executable computer instructions at said client system.

14. The document distribution system of claim 13, further comprising means for compressing said compound document prior to transmitting said compound document.

15. The document distribution system of claim 13, further comprising means for scaling said compound document for display of said document at said client system.

16. The document distribution system of claim 13, wherein said means for receiving a request for transferring said compound document is responsive to a request coded in the HTTP protocol.

17. The document distribution system of claim 13, wherein said pseudo computer instructions comprise JAVA bytecode.

18. The document distribution system of claim 13, wherein said compound document includes information selected from the group of text, image, and graphic data.

19. A document display system, executable on a client system, for displaying at said client system a compound document residing on a server system, said client system having a computer processor therein and capable of communication with said server system, said document display system comprising:

means for requesting said compound document from said server system;

means for receiving from said server system said compound document and first computer instructions indicative of second computer instructions for displaying said compound document at said client system;

means for generating in response to said first computer instructions said second computer instructions computer, said second computer instructions being executable by said processor of said client system for displaying said compound document at said client system; and

means for executing at said client system said second computer instructions, and displaying said compound document at said client system.

20. The document display system of claim 19, wherein said means for executing further includes a Java Virtual Machine for deriving said second computer instructions.

21. The document display system of claim 19, wherein said means for requesting includes requesting said compound document using the HTTP protocol.

22. The document display system of claim 19, wherein said first computer instructions comprise JAVA bytecode.

23. The document display system of claim 19, wherein said means for requesting includes browser software, executable on said client system.

24. The document display system of claim 19, wherein said compound document includes information selected from the group of text, image, and graphic data.

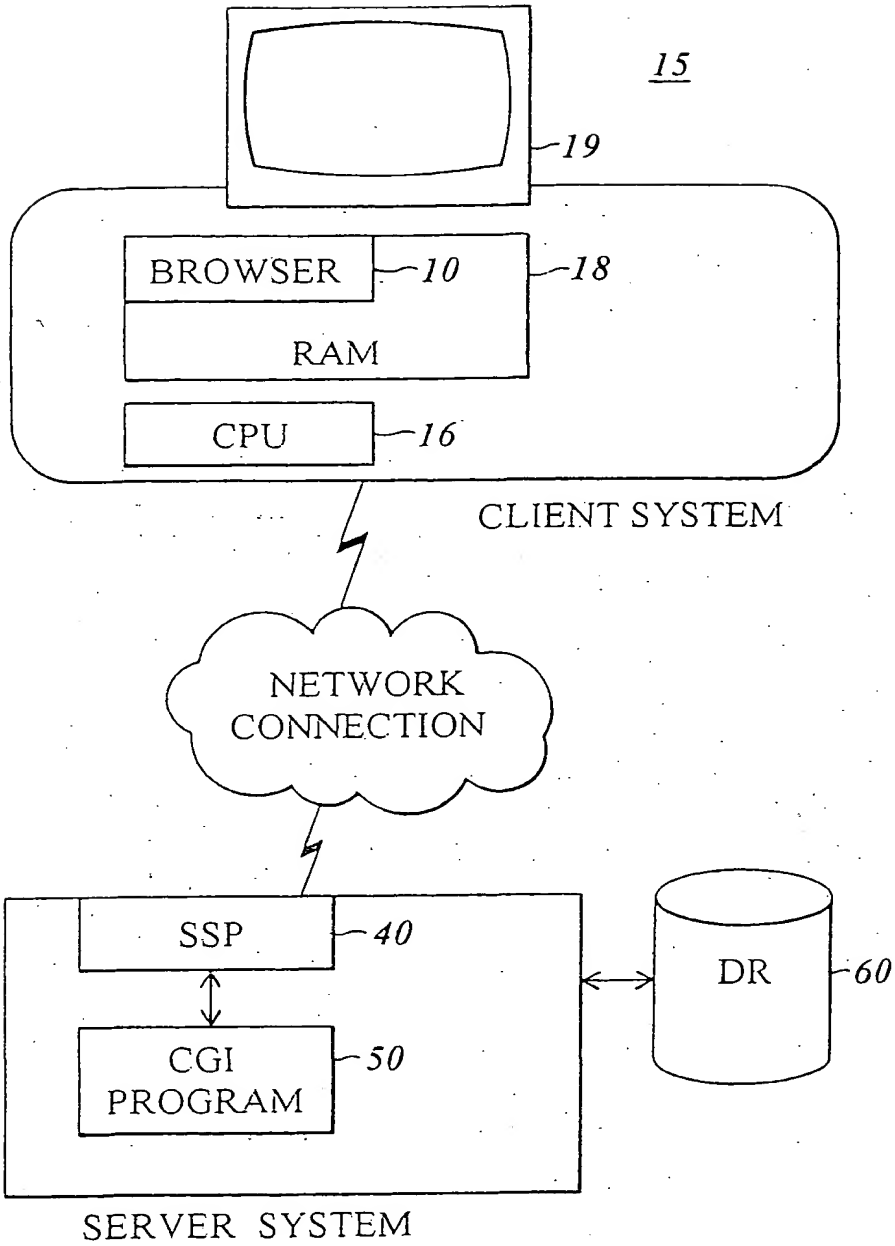

15
19
BROWSER
10
18
RAM
CPU
16
CLIENT SYSTEM
NETWORK CONNECTION
SSP
40
CGI PROGRAM
50
DR
60
SERVER SYSTEM

FIG. 1

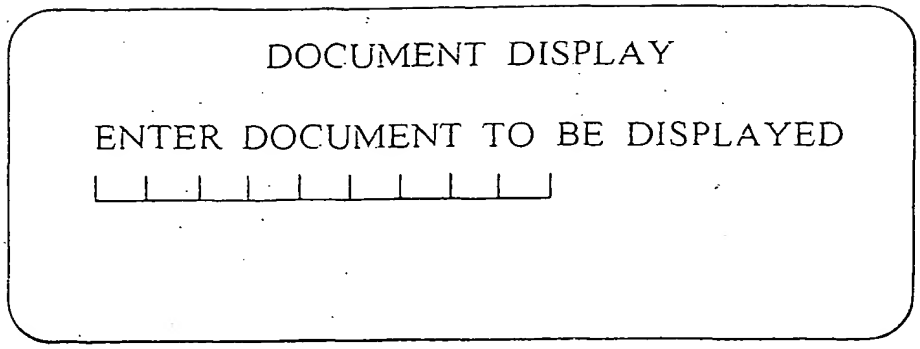

DOCUMENT DISPLAY
ENTER DOCUMENT TO BE DISPLAYED

FIG. 2A

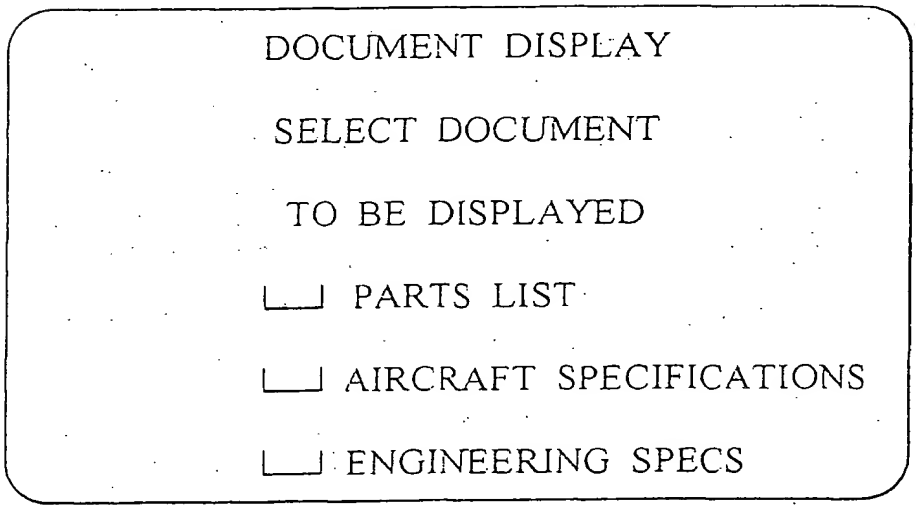

DOCUMENT DISPLAY
SELECT DOCUMENT
TO BE DISPLAYED
PARTS LIST
AIRCRAFT SPECIFICATIONS
ENGINEERING SPECS

FIG. 2B

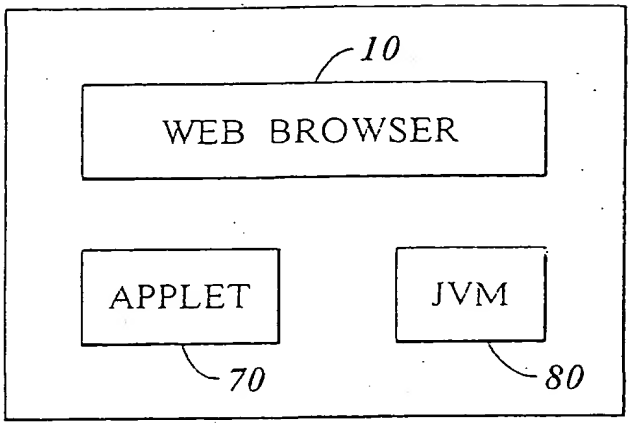

10
WEB BROWSER
APPLET
JVM
70
80

FIG. 3

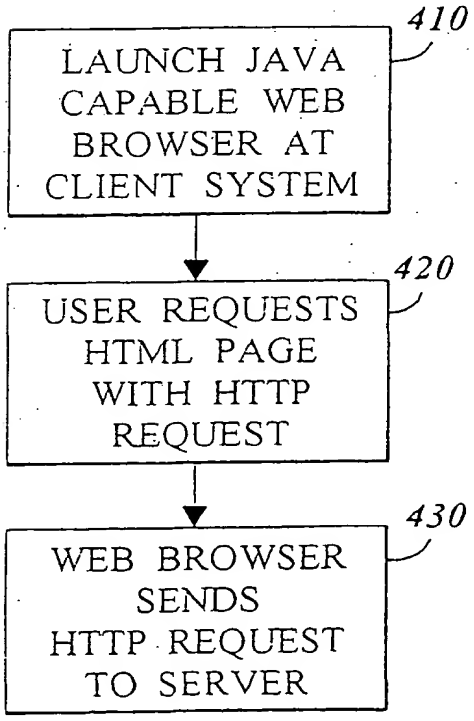

410
LAUNCH JAVA CAPABLE WEB BROWSER AT CLIENT SYSTEM
420
USER REQUESTS HTML PAGE WITH HTTP REQUEST
430
WEB BROWSER SENDS HTTP REQUEST TO SERVER

FIG. 4A

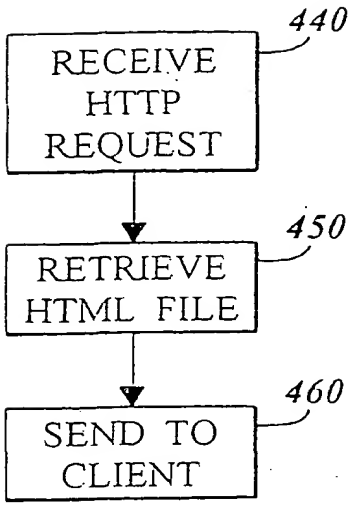

440
RECEIVE HTTP REQUEST
450
RETRIEVE HTML FILE
460
SEND TO CLIENT

FIG. 4B

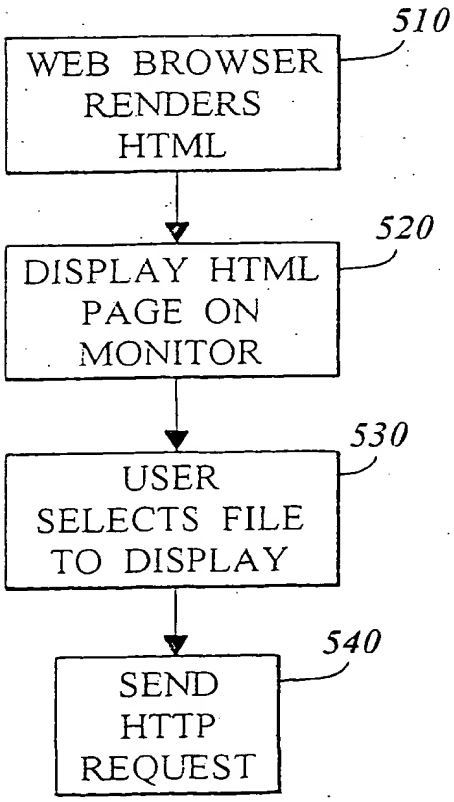

510
WEB BROWSER RENDERS HTML
520
DISPLAY HTML PAGE ON MONITOR
530
USER SELECTS FILE TO DISPLAY
540
SEND HTTP REQUEST

FIG. 5A

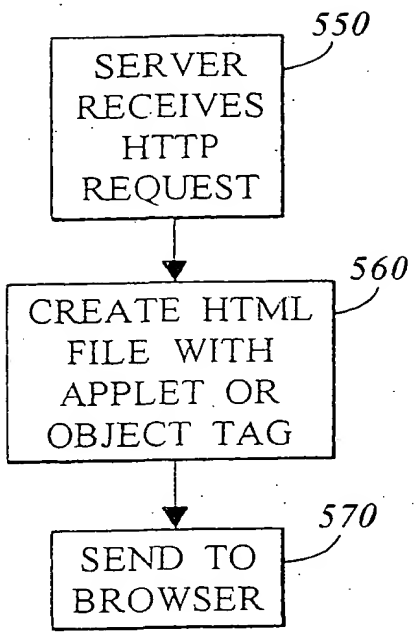

550
SERVER RECEIVES HTTP REQUEST
560
CREATE HTML FILE WITH APPLET OR OBJECT TAG
570
SEND TO BROWSER

FIG. 5B

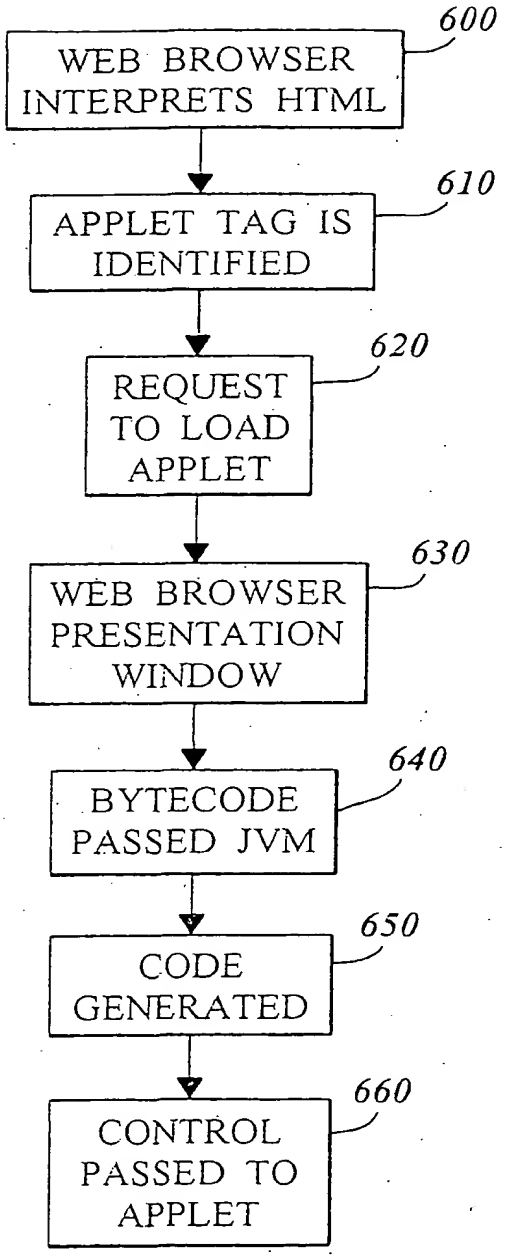

600
WEB BROWSER INTERPRETS HTML
610
APPLET TAG IS IDENTIFIED
620
REQUEST TO LOAD APPLET
630
WEB BROWSER PRESENTATION WINDOW
640
BYTECODE PASSED JVM
650
CODE GENERATED
660
CONTROL PASSED TO APPLET

FIG. 6A

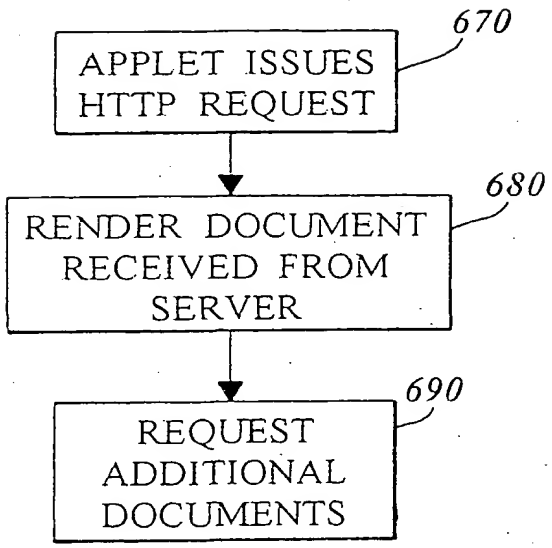

APPLET ISSUES HTTP REQUEST
670
RENDER DOCUMENT RECEIVED FROM SERVER
680
REQUEST ADDITIONAL DOCUMENTS
690

FIG. 6B

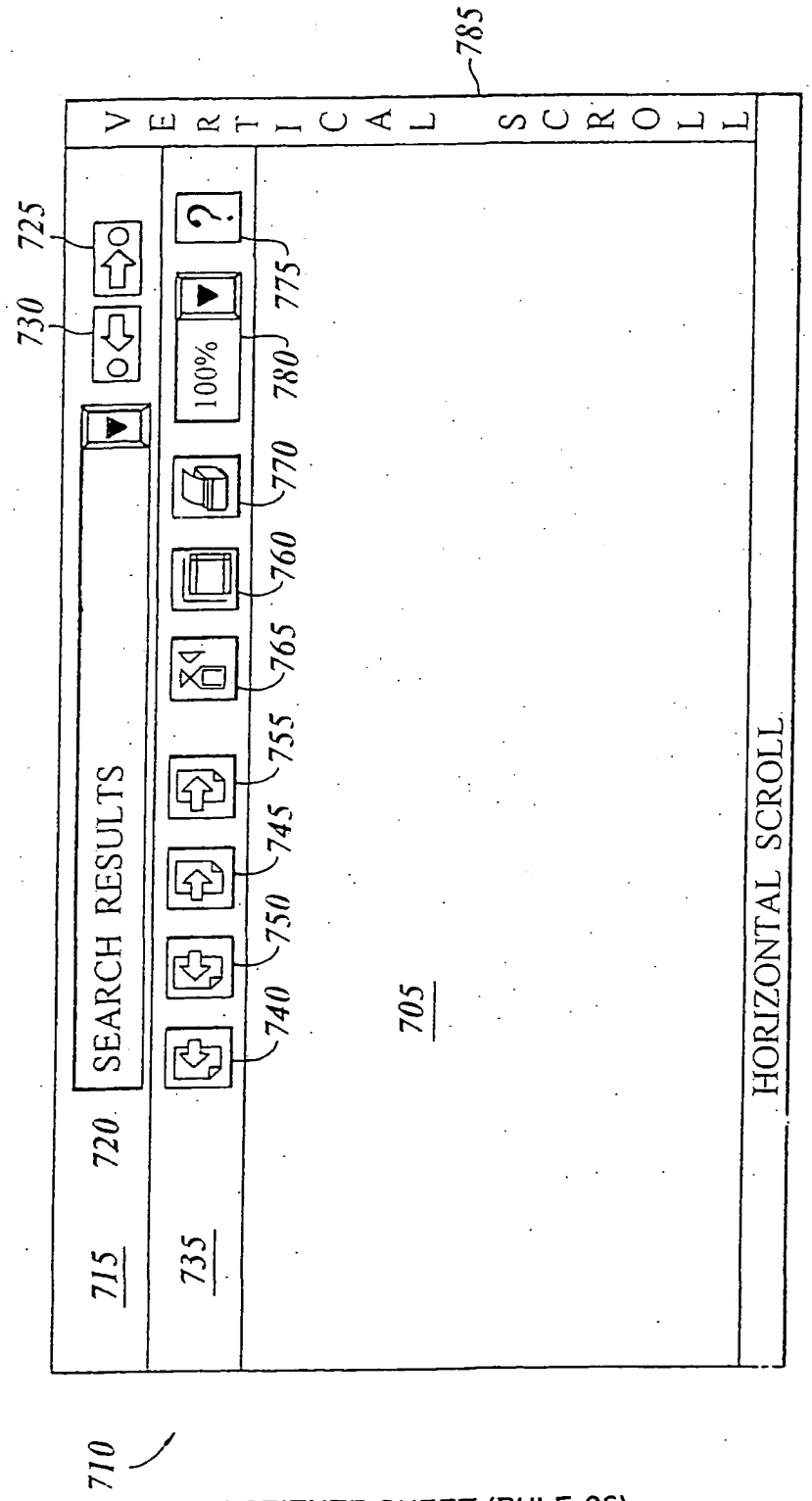

VERTICAL SCROLL
785
725
730
SEARCH RESULTS
100%
775
780
770
760
765
755
745
750
740
715
720
735
705
HORIZONTAL SCROLL
710

FIG. 7

INTERNATIONAL SEARCH REPORT

International Application No
PC., US 99/15091

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/30 G06F9/46

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 809 210 A (XEROX CORP) 26 November 1997 (1997-11-26) page 2, line 34 - line 46; figures 8,24,25 page 20, line 8 - line 33	1-24
X	EP 0 818 742 A (IBM) 14 January 1998 (1998-01-14) the whole document	1-24

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
5 November 1999	12/11/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Fonderson, A

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

nformation on patent family members

inte··· ational Application No
PC./US 99/15091

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0809210 A	26-11-1997	US 5884014 A	16-03-1999
		BR 9703309 A	15-09-1998
		BR 9703310 A	10-11-1998
		CA 2199437 A	23-11-1997
		CA 2205732 A	23-11-1997
		EP 0809192 A	26-11-1997
		JP 10100484 A	21-04-1998
		JP 10097397 A	14-04-1998
EP 0818742 A	14-01-1998	GB 2315140 A	21-01-1998
		JP 10133838 A	22-05-1998